

Sesión 05

Configuración y Resiliencia

Instructor:

ERICK ARÓSTEGUI

earostegui@galaxy.edu.pe



6 NET

MICROSERVICES ARCHITECTURE

ÍNDICE

01 Servidor de configuración

02 Registro y discovery de microservicios.

03 Resiliencia y alta disponibilidad de microservicios.

04 Principales patrones de resiliencia (Circuit Breaker, Retry Design y Bulkheads Design).

01



Servidor de configuración

Configuración de servicios mediante configuración distribuida

¿Qué tiene de diferente administrar la configuración en una aplicación nativa de la nube?

Configuración: no distribuido vs distribuido



**De uno o un puñado
de archivos de
configuración**

Configuración: no distribuido vs distribuido



**De uno o un puñado
de archivos de
configuración**



A...



**Muchos, muchos
archivos de
configuración**

Configuración: no distribuido vs distribuido



Herramientas de gestión de configuración al rescate,
¿verdad?

e.g. Chef/Puppet/Ansible

Configuración: no distribuido vs distribuido

Funcionará ... pero no es ideal en la nube



Configuración: no distribuido vs distribuido



**Orientado al
despliegue**

Configuración: no distribuido vs distribuido



**Orientado al
despliegue**



**Basado en PUSH
generalmente no es lo
suficientemente
dinámico**

Configuración: no distribuido vs distribuido



Orientado al
despliegue



Basado en PUSH
generalmente no es lo
suficientemente
dinámico



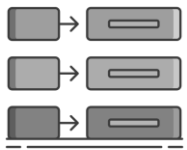
Basado en PULL
agrega latencia con
sondeo temporal

P: Si las herramientas de administración de configuración no resuelven nuestro problema, ¿qué lo hace?

P: Si las herramientas de administración de configuración no resuelven nuestro problema, ¿qué lo hace?

R: Servidor de configuración

Servidor de Configuración de Aplicaciones



Almacén de clave / valor
centralizado, dinámico y
dedicado (puede distribuirse)



Fuente con acceso de
autorización



Revisión de cuentas



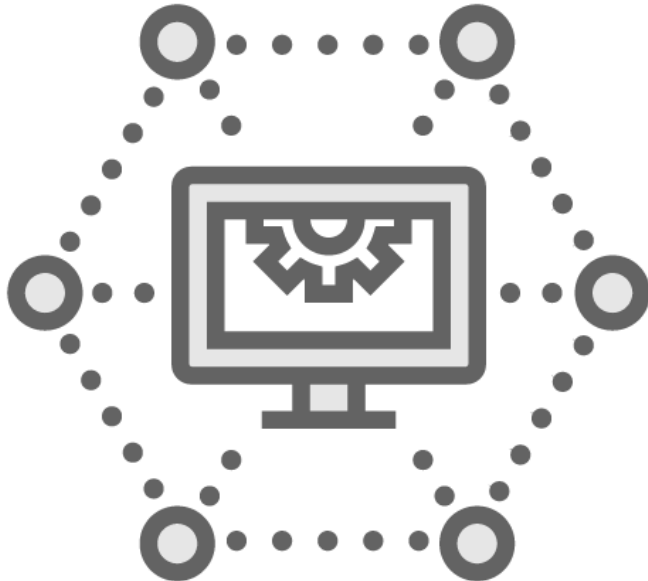
Versionado



Soporte de criptografía

Administrar la configuración de la aplicación con **Spring Cloud**

Administrar la configuración con



- Spring Cloud Consul
- Spring Cloud Zookeeper
- **Spring Cloud Config**

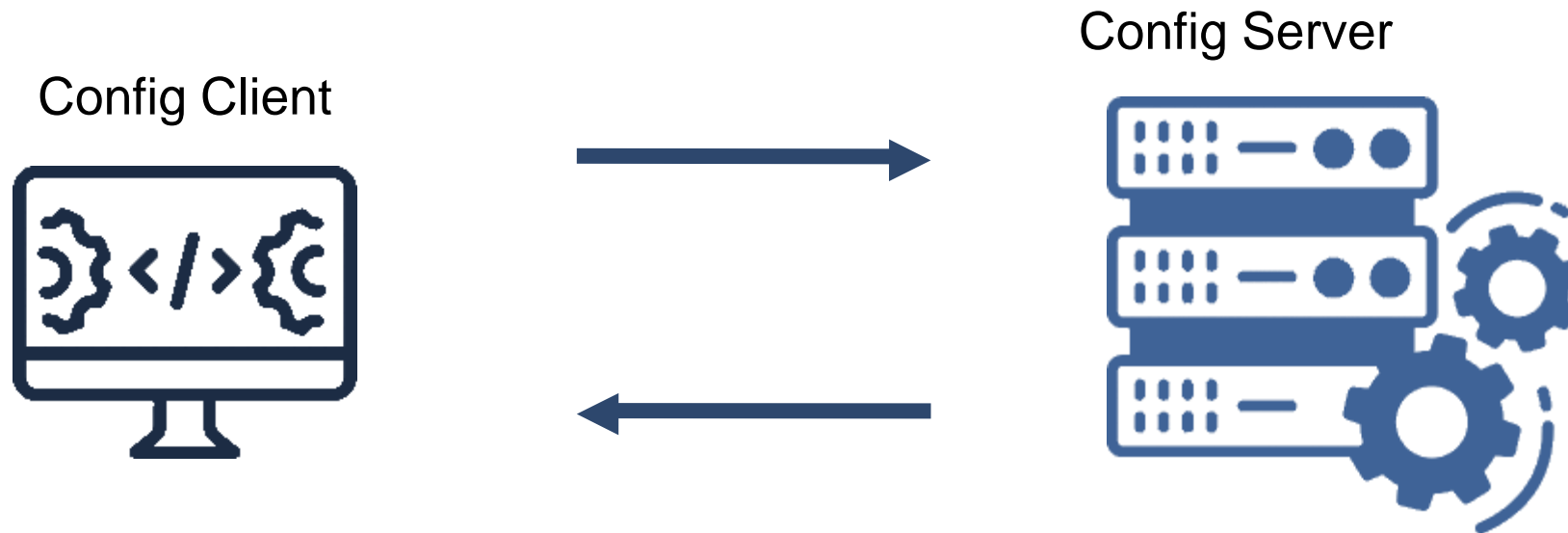
Spring Cloud Config

Spring Cloud Config proporciona soporte del servidor y del lado del cliente para la configuración externa en un sistema distribuido.

Documentación de referencia :

<https://cloud.spring.io/spring-cloud-config/reference/html/>

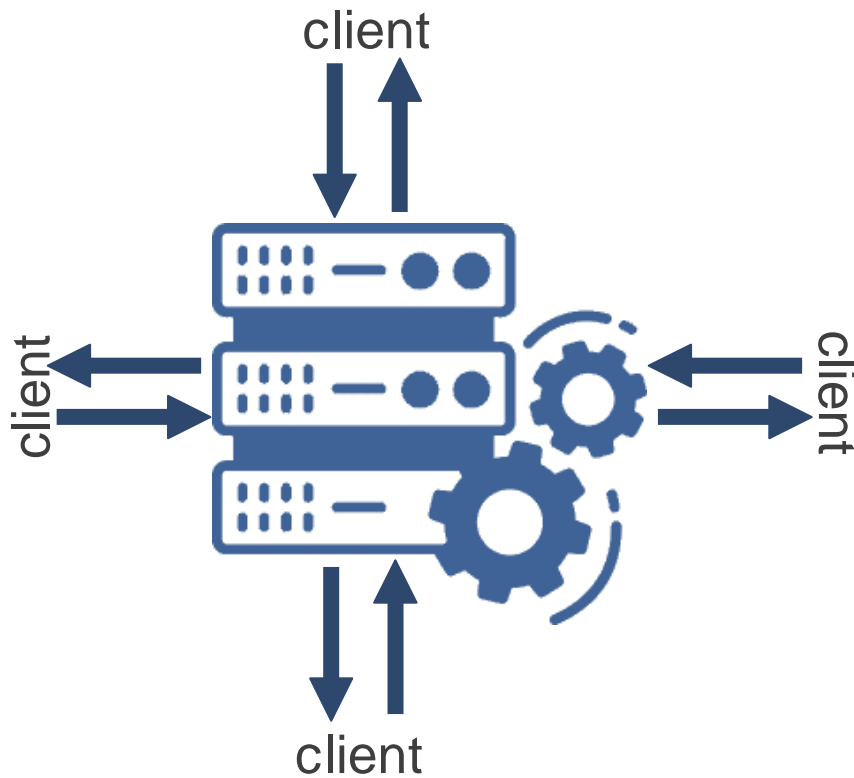
Integración con aplicaciones



- **Embebido en la aplicación**
- **Spring Environment abstraction**
 - e.g. @Inject Environment

- **Standalone (puede embeberse)**
- **Spring PropertySource abstraction**
 - e.g. classpath:file.properties

Spring Cloud Config Server



Acceso HTTP REST

Formatos de salida

- **JSON (default)**
- Properties
- YAML

Almacenamiento Backend

- **Git (default)**
- SVN
- Filesystem

Enviroments de configuración

Spring Cloud Config Server

¡No olvides asegurar tu servidor de configuración!

Fácil de configurar **Spring Security**



REST Endpoint Parameters

{application}

maps to
spring.application.name
on client

{profile}

maps to
spring.profiles.active
on client

{label}

función del lado del
servidor para referirse al
conjunto de archivos de
configuración por nombre

REST Endpoint



Endpoint

GET /{application}/{profile}[/{label}]



Ejemplo

- /myapp/dev/master
- /myapp/prod/v2
- /myapp/default

REST Endpoint



Endpoint

`/ {application}-{profile}.(yml | properties)`

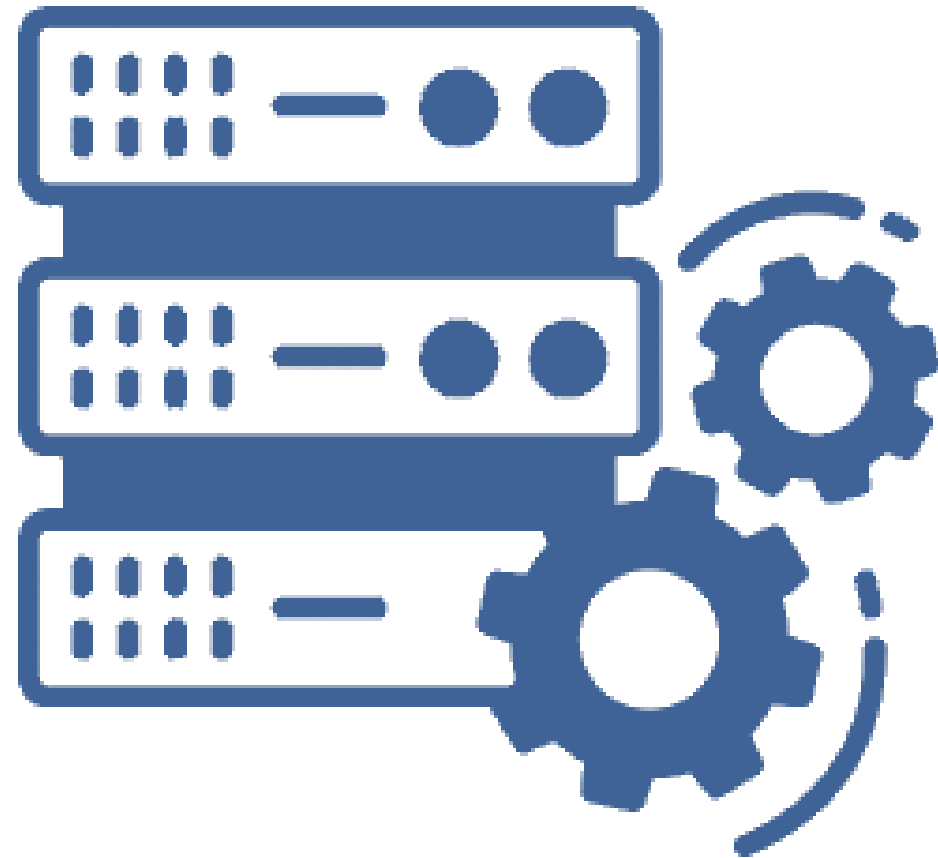


Ejemplo

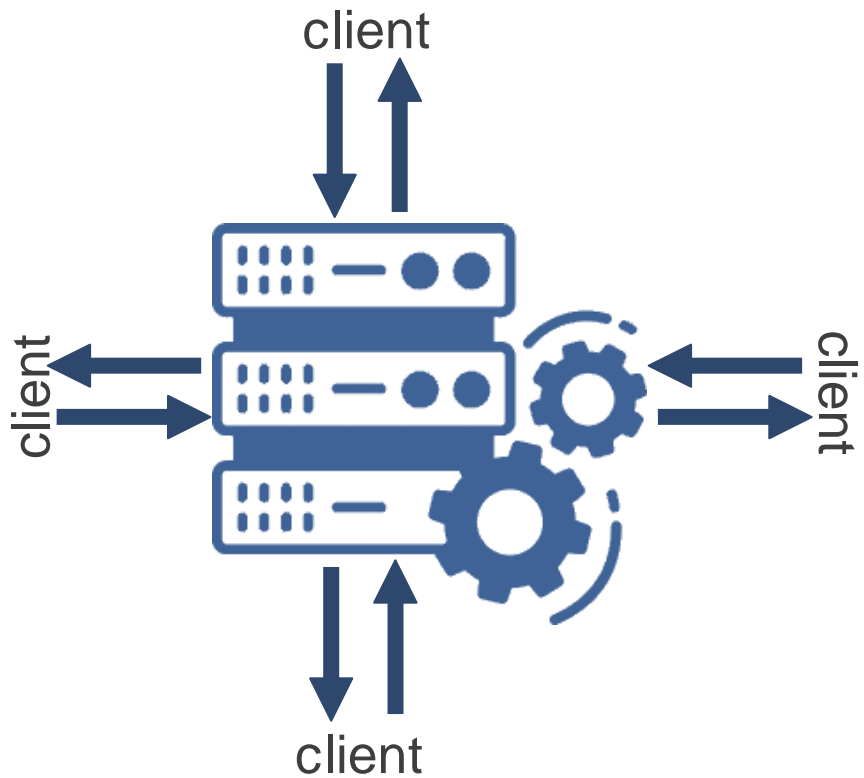
- `/myapp-dev.yml`
- `/myapp-prod.properties`
- `/myapp-default.properties`

Creando e iniziando un config server

DEMO

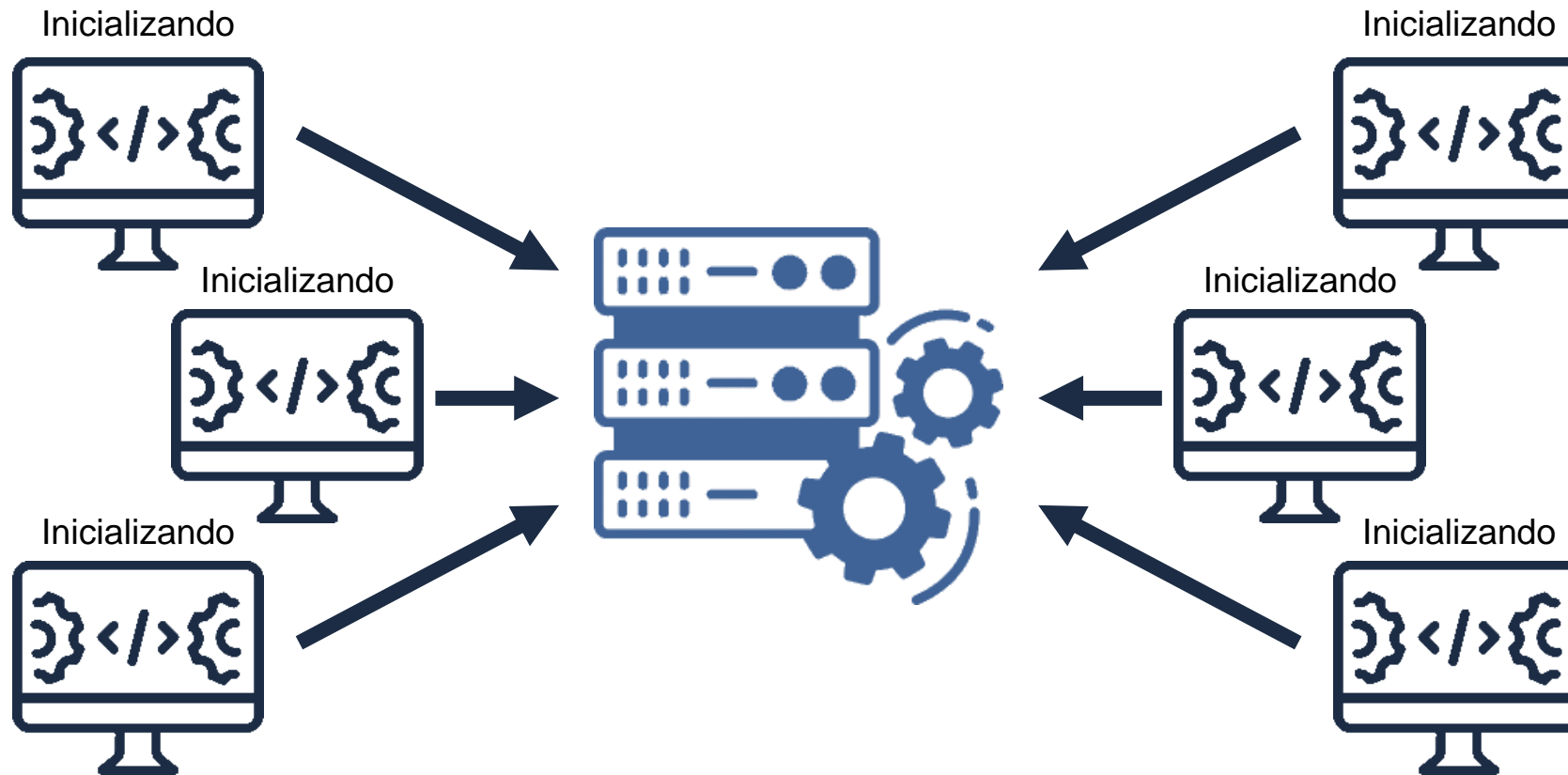


Spring Cloud Config Client

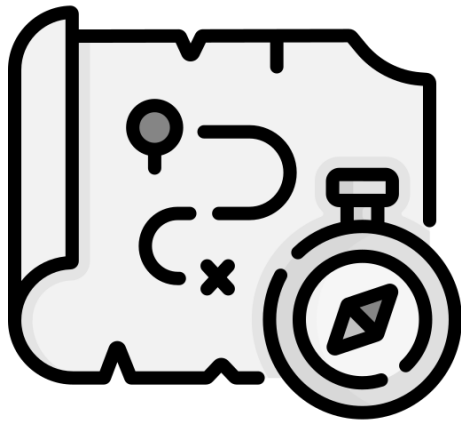


Inicialización y carga de configuración de aplicaciones

Recuperando configuración: inicio de la aplicación



Bootstrapping



Config first

Especifique la ubicación del servidor de configuración



Discovery first

Descubre la ubicación del servidor de configuración

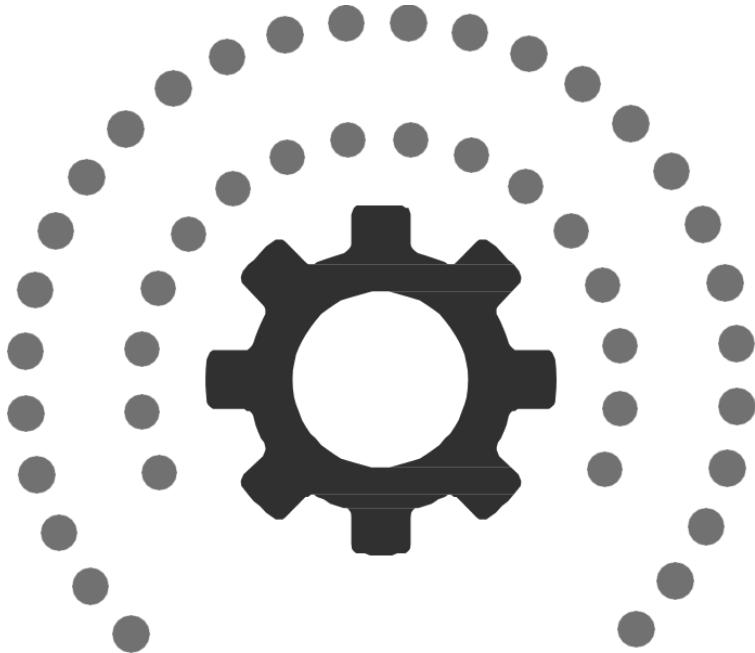
Inicialización de un servicio que usa config client

DEMO



Actualización de la configuración en tiempo de ejecución

Actualización de Config Client

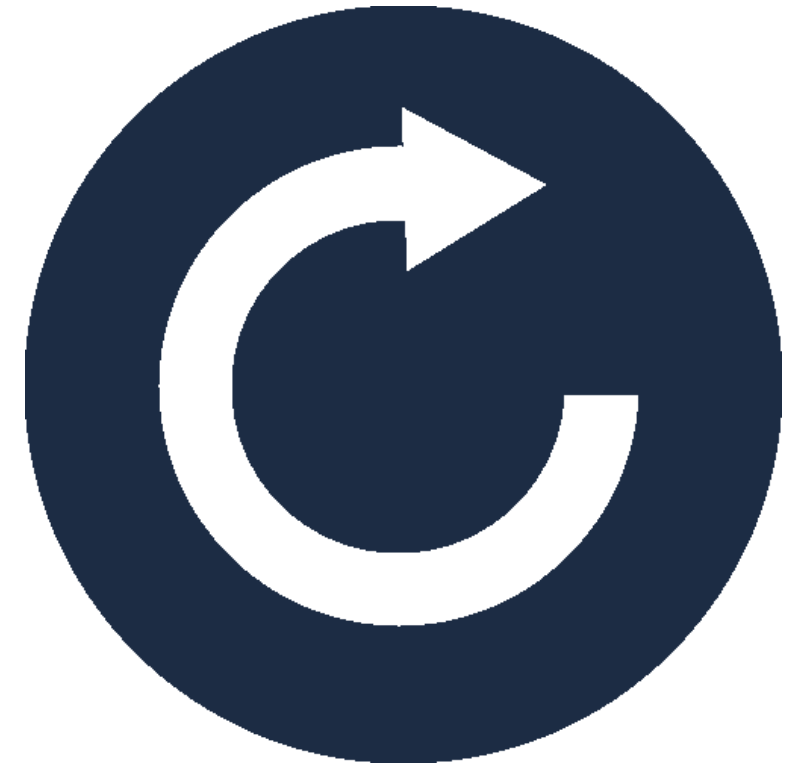


Refresh

@ConfigurationProperties

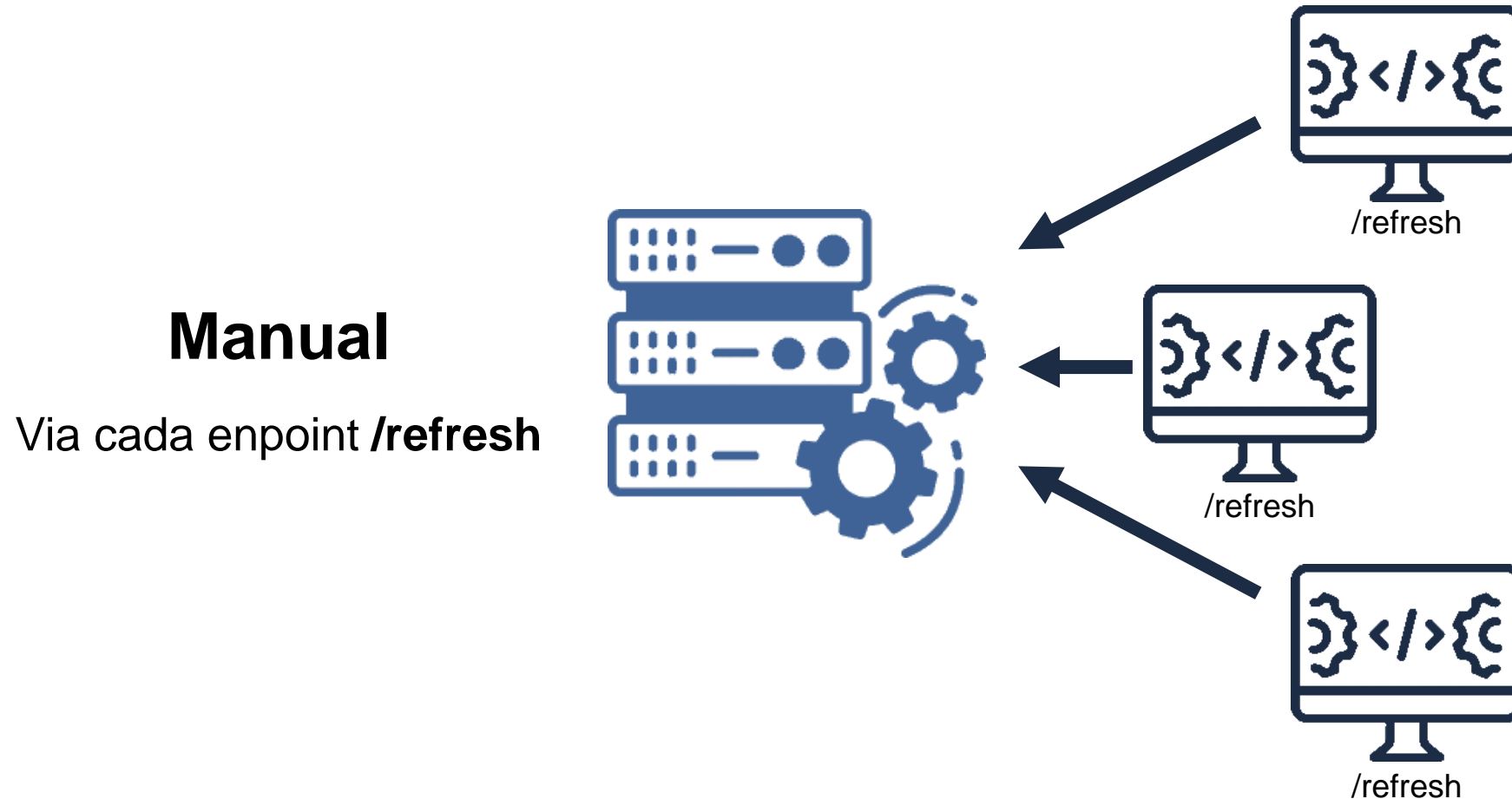
Notifique las aplicaciones para actualizar la configuración

/refresh
with
spring-boot-actuator



→ Servidor de configuración

Obteniendo configuración: actualización explícita



Notifique las aplicaciones para actualizar la configuración



/refresh

with
spring-boot-actuator

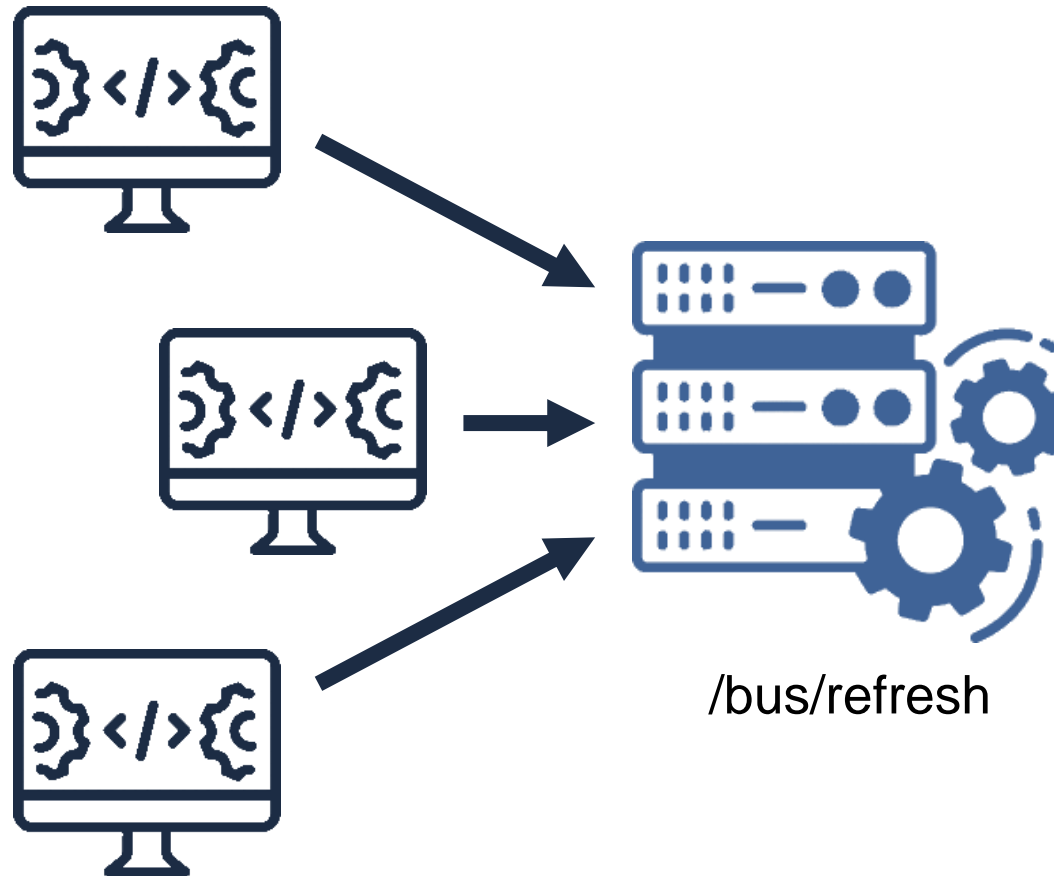


/bus/refresh

with
spring-cloud-bus

→ Servidor de configuración

Recuperación de la configuración: actualización dinámica por push

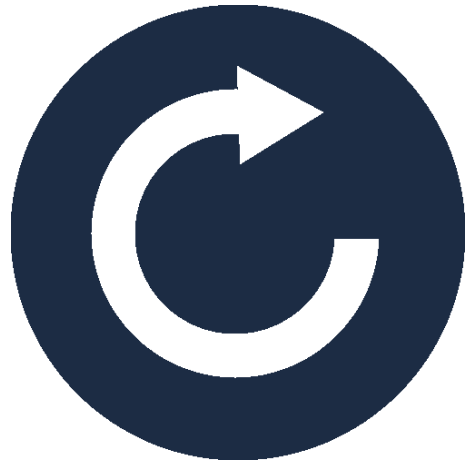


Automático

Via bradcasting Spring

Cloud Bus

Notifique las aplicaciones para actualizar la configuración



/refresh

with
spring-boot-actuator



/bus/refresh

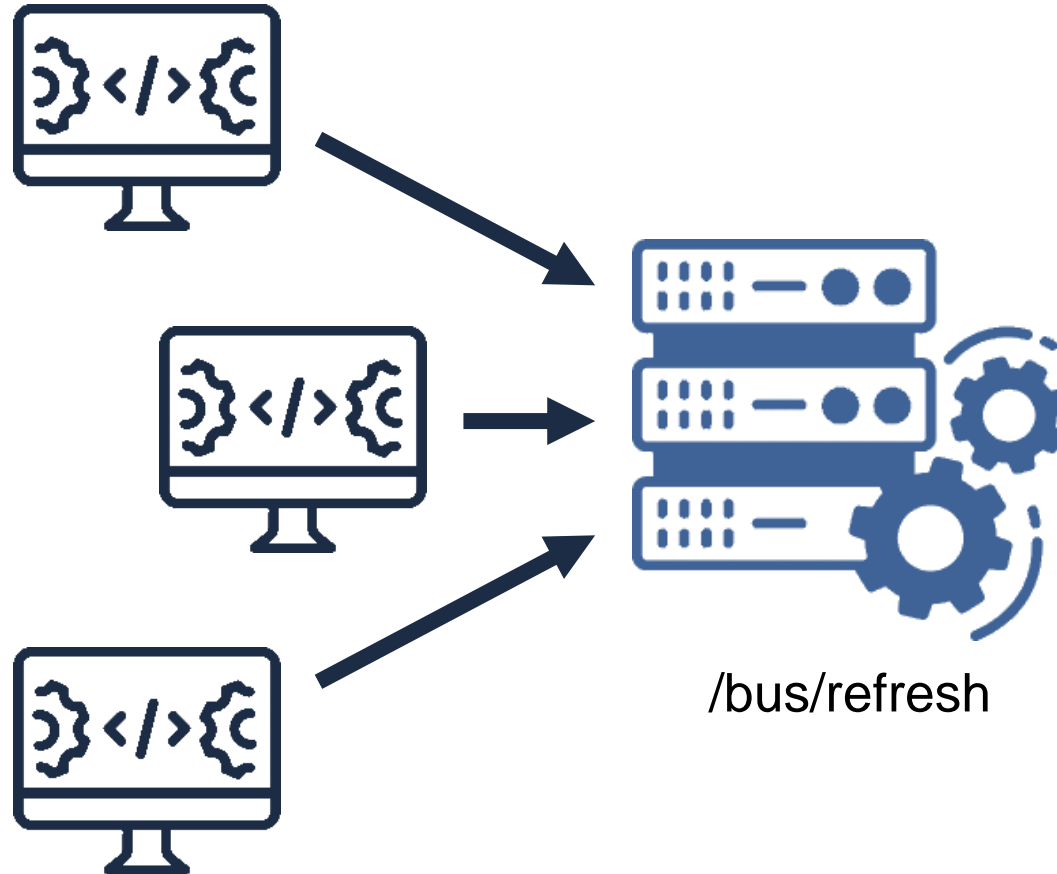
with
spring-cloud-bus



VCS + /monitor

with
spring-cloud-config- monitor &
spring-cloud-bus

Recuperando configuración: Actualización inteligente

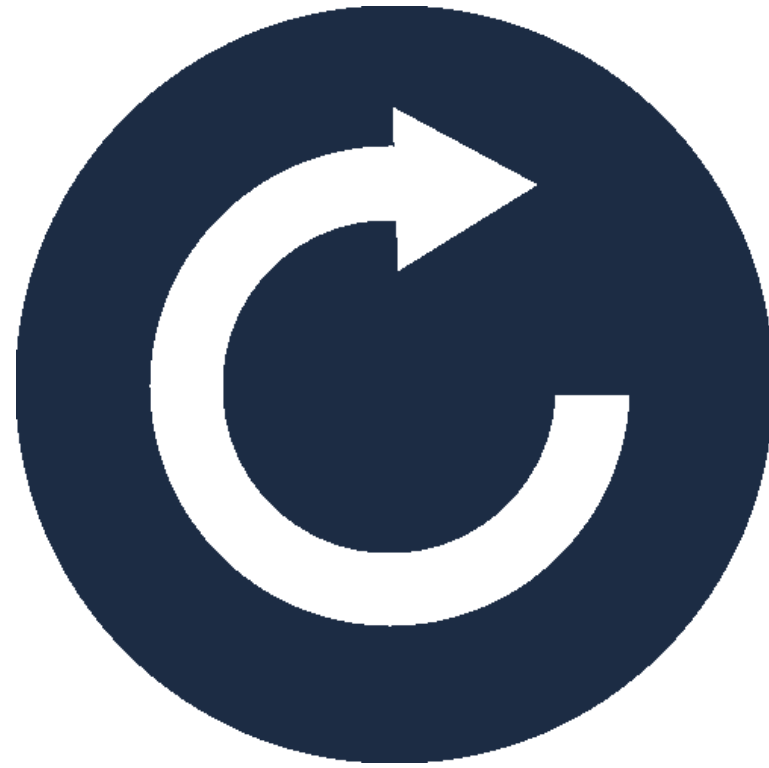


**Automático e
Inteligente**

Via post commit hooks
Spring Cloud Config Monitor
& Spring Cloud Bus
broadcasting

Actualización de configuración al vuelo

DEMO



¿Qué características son compatibles?



Configuración
encriptada en
reposo y / o en
vuelo



Endpoint de
encriptamiento
de
configuración



Endpoint de
descripta
miento de
configuración



Cifrado y descifrado
con claves
simétricas o
asimétricas.

02

Registro y discovery de microservicios.

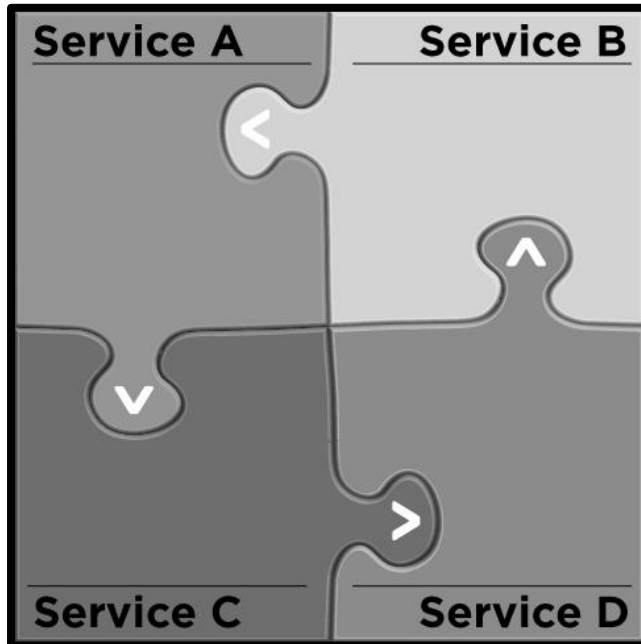


Encontrar servicios utilizando el **Services Discovery**

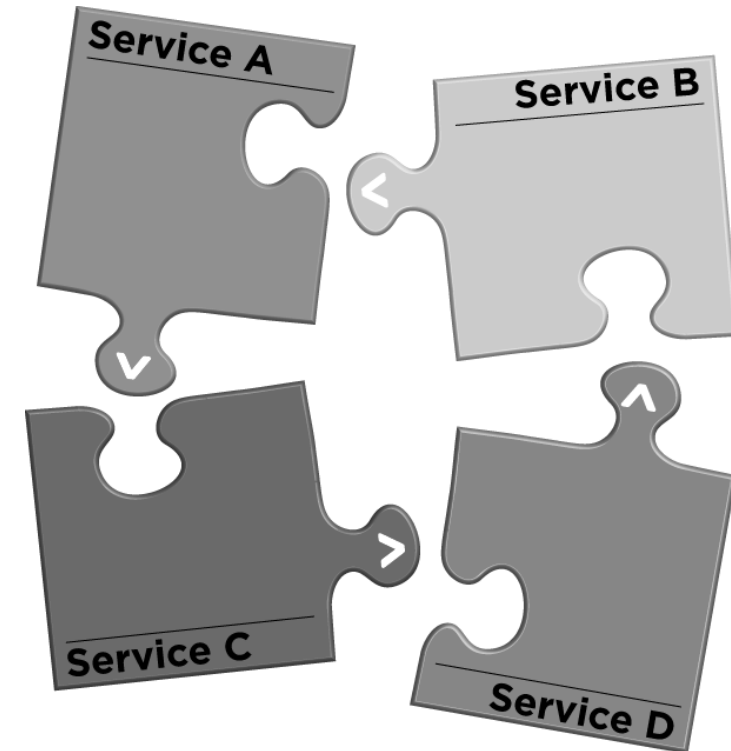
¿Qué es **Service Discovery** y por qué lo necesitamos?

→ Registro y discovery de microservicios

Cambios en la forma en que desarrollamos software



Desde una aplicación
monolítica



Para servicios desplegables
individualmente

El problema: ¿cómo un servicio ubica a otro?



Servicio de
aplicación A

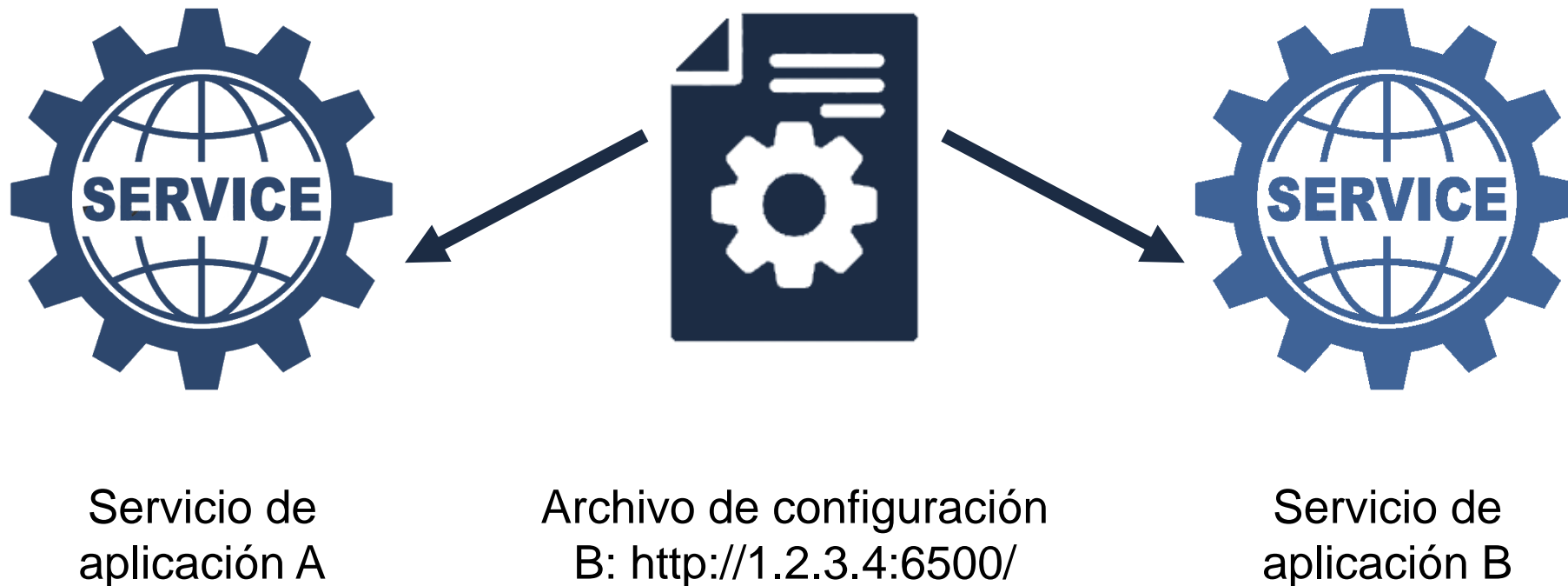


¿Localización?

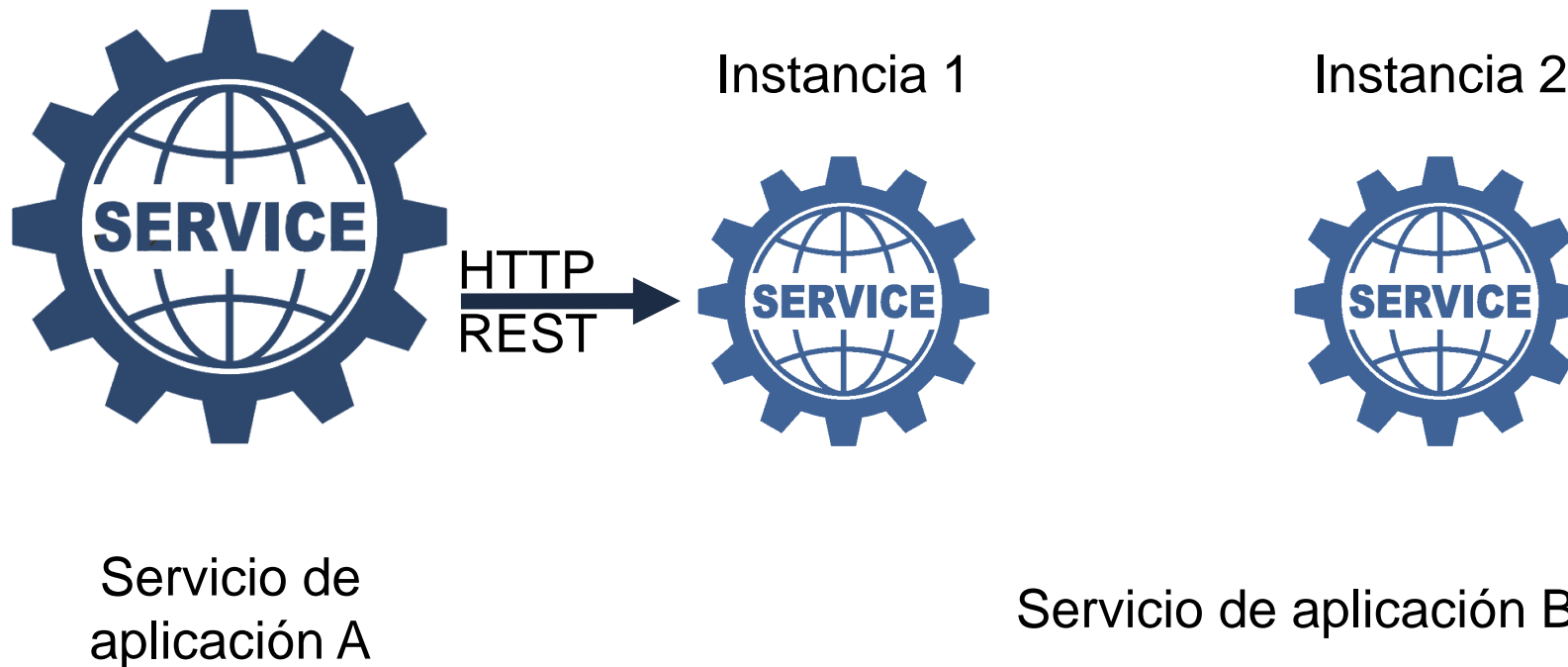


Servicio de
aplicación B

El enfoque simple: a través de la configuración

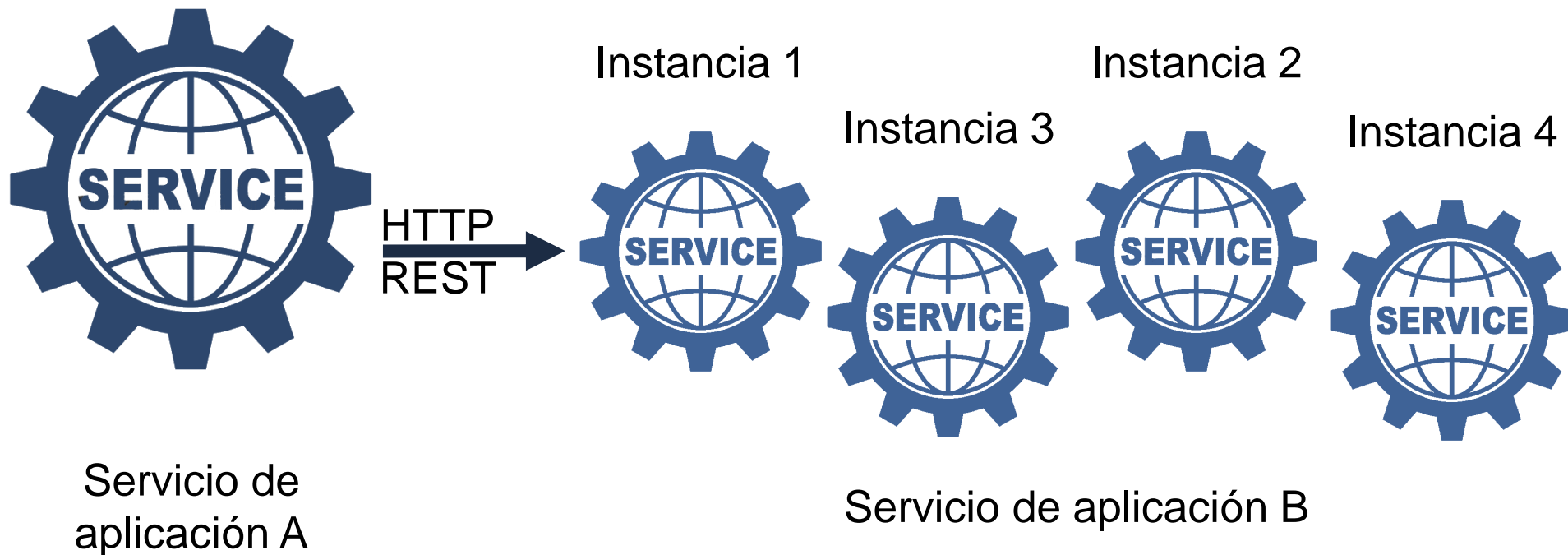


Multiple Instancias

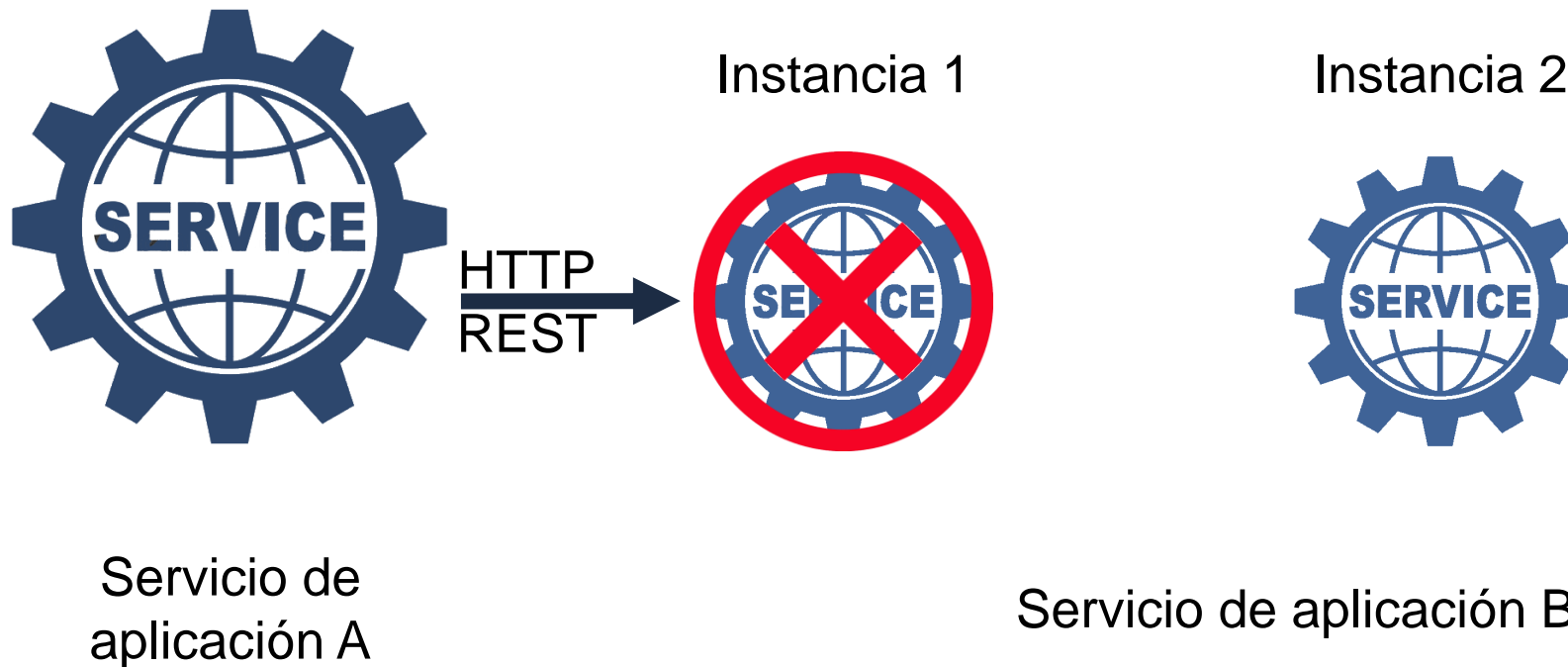


→ Registro y discovery de microservicios

Las instancias van y vienen en respuesta a la demanda



Las instancias fallan



El enfoque simple: a través de la configuración

¡El enfoque simple es
demasiado estático
(congelado en el tiempo)
para la nube!



Service Discovery : a través de la configuración



El descubrimiento de servicios proporciona

- Una forma para que un servicio se registre
- Una forma para que un servicio se desregistre
- Una manera para que un cliente encuentre otros servicios
- Una forma de verificar el estado de un servicio y eliminar instancias caídas

Discovering Services con Spring Cloud

Service Discovery : a través de la configuración

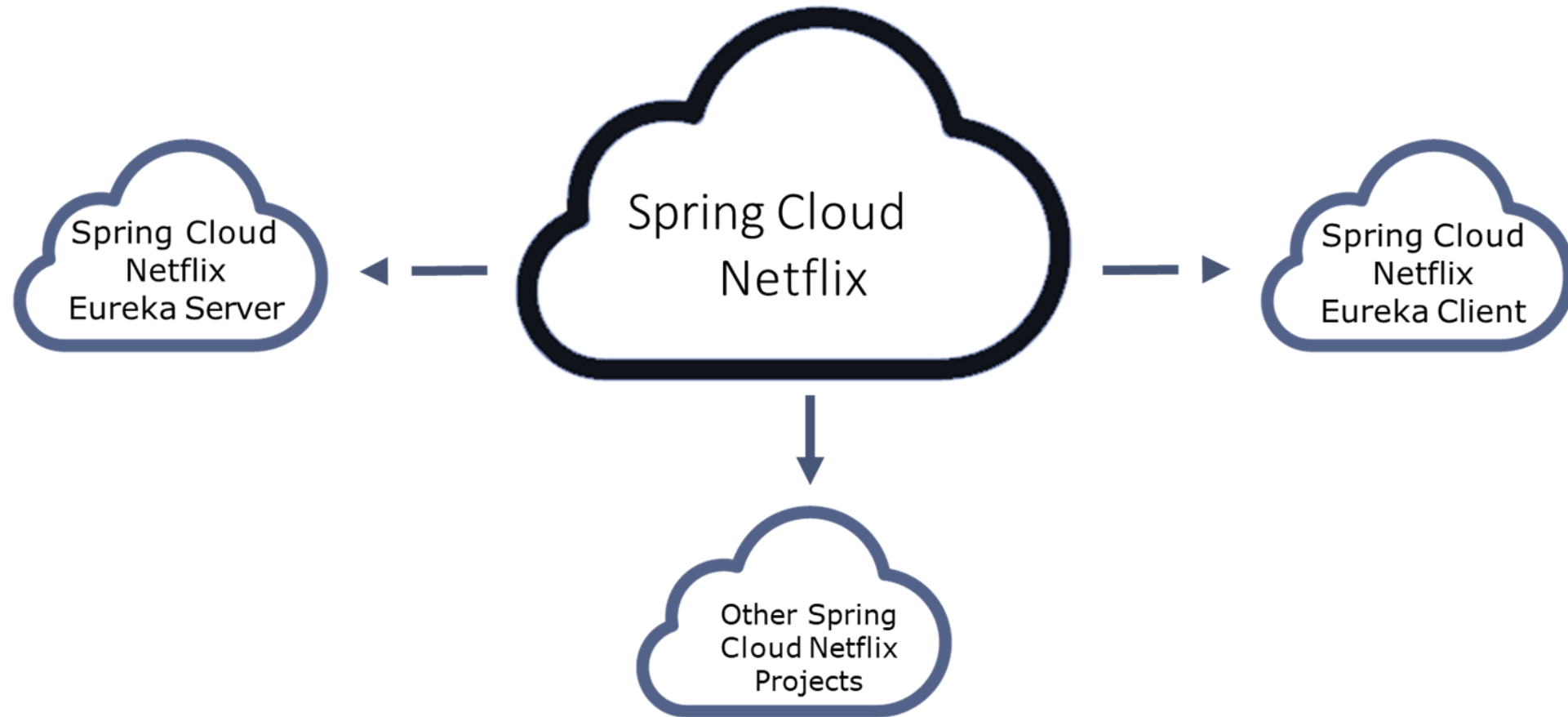


Discover services con:

- Spring Cloud Consul
- Spring Cloud Zookeeper
- Spring Cloud Netflix

Netflix OSS + Spring + Spring Boot
=
Spring Cloud Netflix

→ Registro y discovery de microservicios



Componentes clave en Service Discovery

Discovery Server



Service

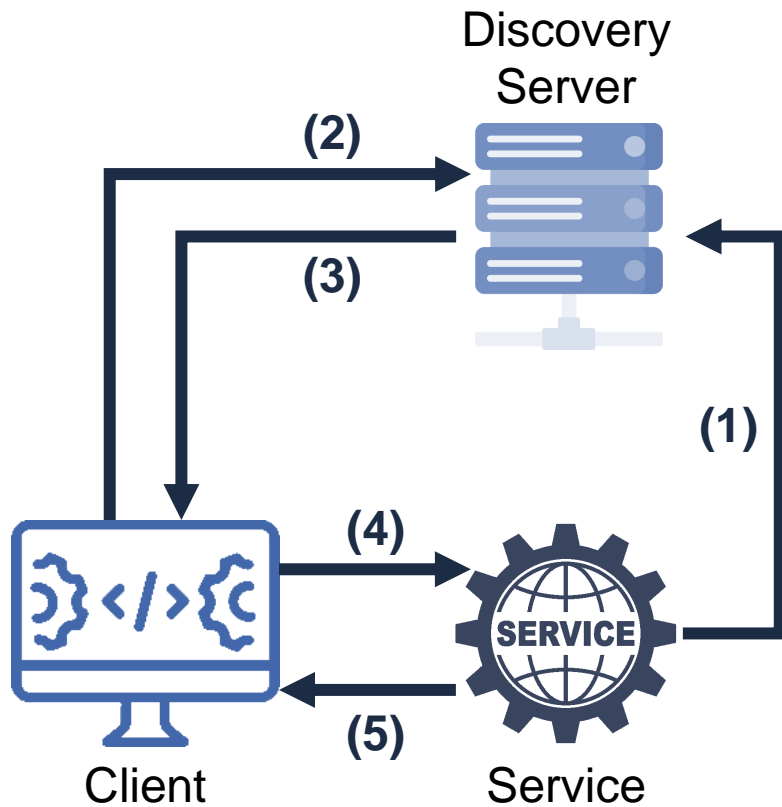


Client



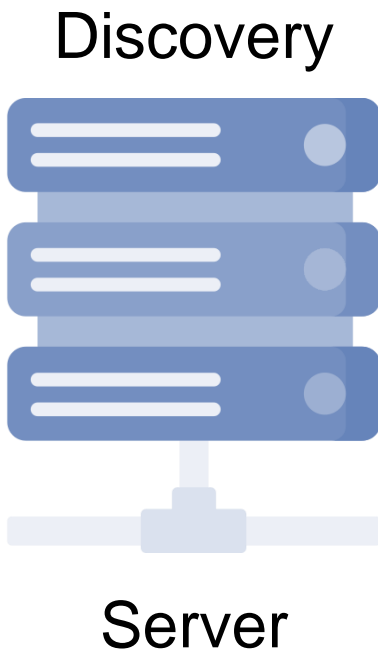
→ Registro y discovery de microservicios

Service Discovery : a través de la configuración



1. El servicio registra la ubicación
2. El cliente busca la ubicación del servicio
3. El servidor de descubrimiento devuelve la ubicación
4. El cliente solicita servicio en la ubicación
5. El servicio envía respuesta

Service Discovery : a través de la configuración



Un registro de ubicaciones de servicio gestionado activamente

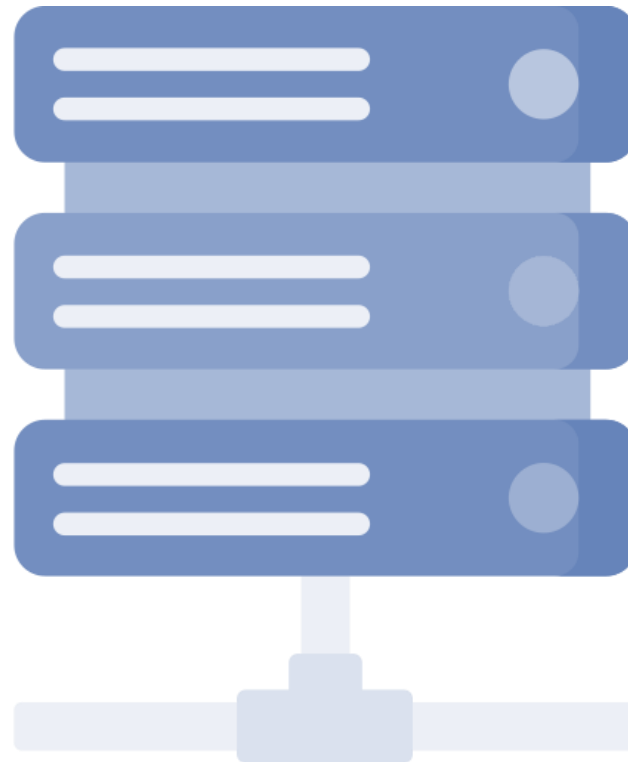
Un solo origen para una o más instancias

Proyecto Spring Cloud:

- **Servidor Spring Cloud Eureka**

Creando e iniziando un discovery server

DEMO



Servicio de que se conecta a un Discovery Server

Application



Service

- Proporcionar algunas funciones de la aplicación.
- El receptor de request
- Una dependencia de otros servicios de una o más instancias
- Uso del **discovery client**
 - Registrarse
 - Darse de baja

Cliente que consume a un Discovery Server

Application

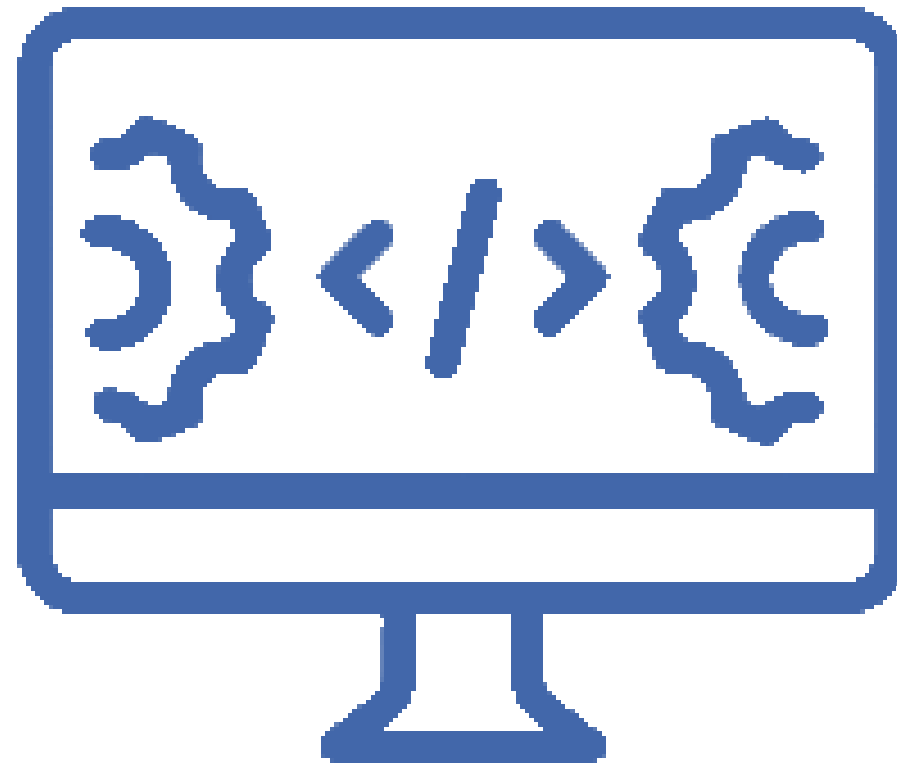


Client

- Llama a otro servicio de aplicación para implementar su funcionalidad
- El emisor de las solicitudes depende de otros servicios Usuario del cliente de descubrimiento
- Encuentra ubicaciones de servicio

Creando un cliente que puede describir servicios

DEMO



Spring Cloud Eureka Dashboard

Cliente que consume a un Discovery Server

Application



Client

Habilitado por defecto

- **`eureka.dashboard.enabled=true`**

Muestra metadatos útiles y estado del servicio

Servidor Eureka: ¿Estan saludables mis servicios?



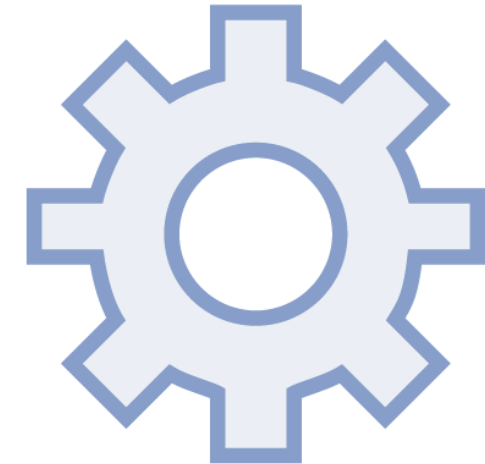
Comprueba
regularmente el
estado de los
servicios.



Los clientes envían
latidos cada 30
segundos
(predeterminado)

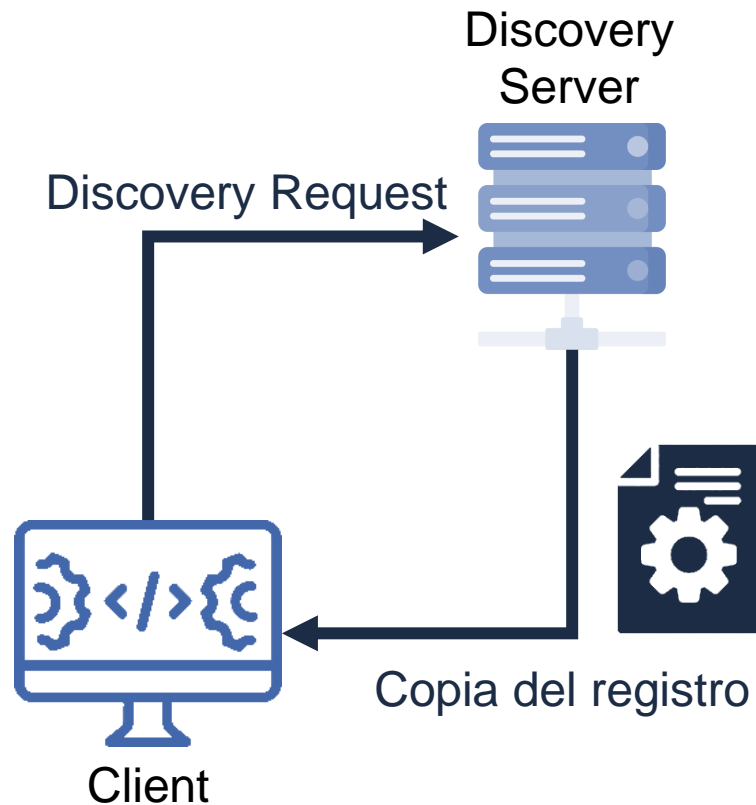


Servicios eliminados
después de 90
segundos sin latidos
(predeterminado)



Puede personalizar la
configuración para
usar
/health endpoint
eureka.client.
healthcheck.enable

Servidor Eureka: ¿Estan saludables mis servicios?



1. El registro se distribuye (almacena en caché localmente en cada cliente)
2. Los clientes pueden operar sin servidor de descubrimiento
3. Obtiene deltas para actualizar el registro

Spring Cloud Eureka AWS Support

Spring Cloud Eureka AWS Support

AWS-specific instance data

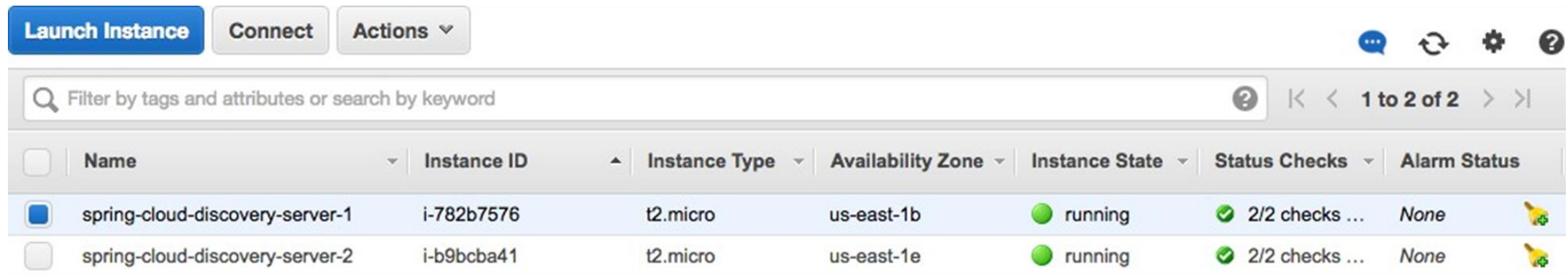
Multi-zone aware

Multi-region aware

Elastic IP Binding

EC2 Dashboard

us-east-1



<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
<input checked="" type="checkbox"/>	spring-cloud-discovery-server-1	i-782b7576	t2.micro	us-east-1b	● running	✓ 2/2 checks ...	None
<input type="checkbox"/>	spring-cloud-discovery-server-2	i-b9bcba41	t2.micro	us-east-1e	● running	✓ 2/2 checks ...	None

```
eureka.client.availability-zones.us-east-1=us-east-1b,us-east-1e
```

Availability Zones Configuration in `application.properties`

```
eureka.client.availability-zones.[region]=[az1],[az2],[az3]
```

Elastic IP Dashboard

us-east-1

Allocate New Address

Actions ▾

↺

⚙

?

Filter

VPC addresses ▾

🔍

Search Elastic IPs...

✕

⏪ < 1 to 2 of 2 Elastic IPs > ⏩

<input type="checkbox"/>	Address	Allocation ID	Instance ID	Network Interface ID	Scope	Private Address
<input type="checkbox"/>	34.192.167.121	eipalloc-bdf9e782	i-782b7576	eni-2d3231d2	vpc	172.31.52.243
<input type="checkbox"/>	34.193.24.166	eipalloc-59e3fd66	i-b9bcba41	eni-f1e0c418	vpc	172.31.33.117

```
eureka.client.service-url.us-east-1b=
```

```
http://ec2-34-192-167-121.compute-1.amazonaws.com:8761/eureka/
```

```
eureka.client.service-url.us-east-1e=
```

```
http://ec2-34-193-24-166.compute-1.amazonaws.com:8761/eureka/
```

Service URL Configuration in `application.properties`

```
eureka.client.service-url.[zone]=http://[eip-dns]/eureka
```

- *Use EIP DNS name. Do not use IP (as of version Eureka 1.4)

Eureka Dashboard: AWS Multi-zone Discovery Servers

DS Replicas

ec2-34-192-167-121.compute-1.amazonaws.com

ec2-34-193-24-166.compute-1.amazonaws.com

Instances currently registered with Eureka

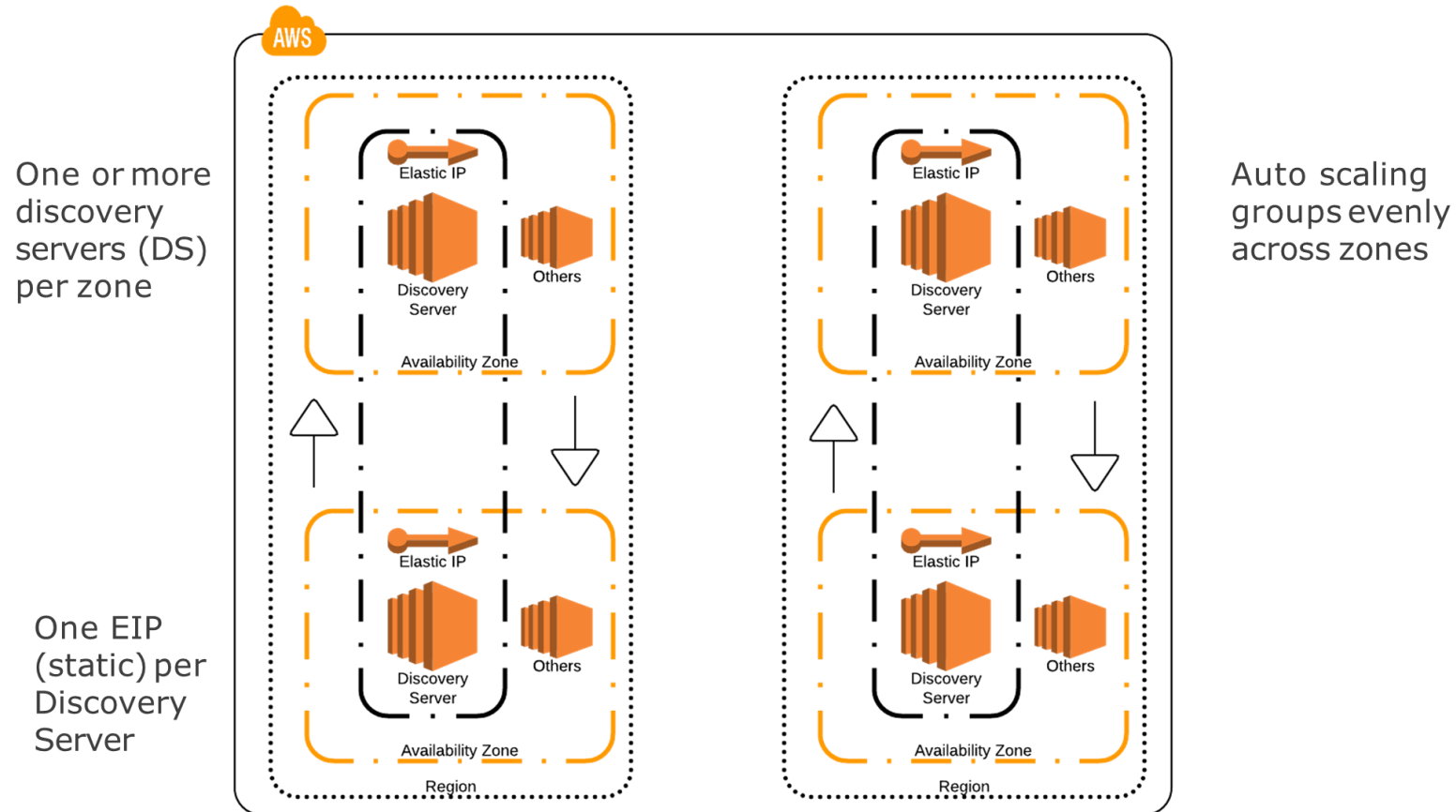
Application	AMIs	Availability Zones	Status
DISCOVERY-SERVER-1	ami-40d28157 (1)	us-east-1b (1)	UP (1) - i-782b7576
DISCOVERY-SERVER-2	ami-40d28157 (1)	us-east-1e (1)	UP (1) - i-b9bcba41

Eureka Dashboard: AWS Instance Data

Instance Info	
Name	Value
public-ipv4	34.192.167.121
public-hostname	ec2-34-192-167-121.compute-1.amazonaws.com
instance-id	i-782b7576
instance-type	t2.micro
ami-id	ami-40d28157
ipAddr	172.31.52.243
status	UP
availability-zone	us-east-1b

→ Registro y discovery de microservicios

Eureka Dashboard: AWS Instance Data



A man with a beard, wearing a striped shirt, is sitting at a desk. He is looking at a laptop screen and has a pen in his mouth. The background is dark blue with a large, light blue circular shape behind the man.

GRACIAS

POR SU PREFERENCIA

